Supporting Software Evolution through Model-Driven Program Transformation

Jing Zhang (Advisor: Jeff Gray)
Department of Computer and Information Sciences – The University of Alabama at Birmingham
http://www.cis.uab.edu/zhang

**BACKGROUND: DOMAIN-SPECIFIC MODELING**

**PROJECT OBJECTIVES**

This project represents an investigation into two research objectives for supporting software evolution in Model-Driven Software Development (MDSD) through Model-Driven Program Transformation (MDPT):

- A generalization process for supporting evolution of legacy software through model-driven program transformation techniques.
- Evolution of model interpreters in the presence of meta-model schema changes.

This project will assist in elevating models and model transformations to first-class development artifacts and facilitate further the ability to evolve model interpreters, as well as a generalized process for transforming legacy software using model-driven techniques.

**TWO-DIMENSIONS OF TRANSFORMATION/TRANSLATION**

**Horizontal transformation**

- Transformation within the same abstraction level (e.g., model transformation, code refactoring).

**Vertical translation**

- Translation or synthesis, between abstraction layers (e.g., code, interpreters, reverse engineering).

**Consequence:** Vertical transformation is needed!

**KEY CHALLENGES**

**Challenge 1:** Application of Model-Driven Software Development to Legacy Software Evolution

- Hard to maintain the fidelity between the model and the legacy source code.
- No support for parsing and interactively transforming legacy source code from higher-level models.

**Challenge 2:** Evolution of Model Interpreters in the Presence of Meta-model Schema Change

- Each evolution of the meta-model breaks the interpreter that was defined on the previous version.
- Changes to the API of the modeling tool may also necessitate adaptations to model interpreters.

**APPROACH 1: MODEL-DRIVEN PROGRAM TRANSFORMATION (MDPT)**

**MDPT for Supporting Legacy System Evolution**

- Based on the unification of a program transformation system with a meta-modeling environment, specifically, unifying the Design Maintenance System (DMS) with GME.
- Key approach: the construction of model interpreters that generate DMS transformation rules from the model specifications.
- Along with the base implementation of a large application, the rules can be fed into DMS to invasively modify a large cross-section of the legacy source.

**Benefits**

- Ensures causal connection between model changes and the underlying source code of the legacy system.
- Allows in legacy evolution from new properties specified in models.
- Model interpreters generate transformation rules to modify source.

**APPROACH 2: MODEL INTERPRETER EVOLUTION ARCHITECTURE (MIEA)**

**Goal: Automation of Model Interpreter Evolution**

- As the meta-model evolves, a formal specification is defined to represent each step of the model transformation.
- The specification language consists of two parts: 1) a pattern description of the interpreter signature; 2) the replacement rule to perform the transformation.
- The transformation specification will be used to generate the DMS rewriting rules to transform the original interpreter into a new one that matches the changes to the meta-model.
- Model Interpreter Evolution in the presence of Modeling Tool API changes.

**CASE STUDY: SUPPORTING A BLACK BOX DATA RECORDER IN A LARGE AVIONICS SYSTEM**

**Description**

In avionics systems, a black box recorder is specified by a logging policy in the source program. This case study demonstrates the process of automatic adaptation of log statements into large source adaptations driven by the model-driven program transformation approach.